# Matching in Multi Agent Pathfinding using M*

**Jonathan Dönszelmann**[1] , **Jesse Mulderij**[1] , **Mathijs de Weerdt**[1]

[1]TU Delft
jdonszelmann@student.tudelft.nl
{m.m.deweerdt, j.mulderij}@tudelft.nl

## Abstract

Todo

## Introduction

A large number of real-world situations require the planning of collisionless routes for multiple agents. For example, the routing of trains over a rail network [1], directing robots in warehouses [2], or making sure autonomous cars do not collide on the road [3]. Problems of this nature are called *Multi agent pathfinding* problems, which in this paper will often be abbreviated as *MAPF*. Solving *MAPF* problems has been proven to be **PSPACE-hard** [4].

One algorithm to solve *MAPF* is called *M** [5]. A standard *A** algorithm as described by Standley [6] plans agents together. This means that in each timestep, the number of possible next states grows exponentially with the number of agents. In *M**, agents follow an individually optimal path, and in each timestep, only the subset of agents which is part of a collision is jointly planned.

A related problem to *MAPF* is the Task Assignment and Pathfinding problem (often abbreviated as *TAPF*). In *TAPF*, agents are grouped into teams. Each team has the same number of goals as the team is large. Which agent ends up on which goal position does not matter. Algorithms solving *TAPF* need to find a *matching* between agents and goal positions of the same team, which produces the shortest paths for all agents. Essentially, *TAPF* is an extention of *MAPF* with the addition of matching. From now on, this problem will be referred to as *MAPFM*.

In this paper, *MAPFM* will be defined, and then it will be investigated if it's possible to extend *M** to solve *MAPFM* problems. To do this, two methods will be proposed. These two methods will be compared, both to each other, and to a number other algorithms solving *MAPFM*. As well as this comparison, a number of extensions to *M** will be investigated applied to both *MAPFM*, and regular *MAPF* problems to improve the runtime performance of *M**.

## I. Problem definition

Stern [7] defines the Multi Agent pathfinding problem as follows:

$$\langle G, s, g \rangle$$

- $G$ is a graph $\langle V, E \rangle$
  - $V$ is a set of vertices
  - $E$ is a set of edges between vertices
- $s$ is a list of $k$ vertices where every $s_i$ is a starting position for an agent $a_i$
- $g$ is a list of $k$ vertices where every $g_i$ is a target position for an agent $a_i$

Though algorithms presented in this paper would work on any graph $G$, in most examples given, $G$ is simplified to be a 4-connected grid (sometimes called a *map* in this paper).

In this paper, this definition of *MAPF* is expanded with matching. The resulting problem is called *MAPFM*, and has the following definition:

$$\langle G, s, g, sc, gc \rangle$$

- $sc$ is an array of colours $sc_i$ for each starting vertex $s_i$
- $gc$ is an array of colours $gc_i$ for each target vertex $g_i$

In *MAPFM*, agents travel from start locations to goal locations (just like in *MAPF*). However, an agent's goal vertex is any goal with the same colour as the agent's start vertex.

Vertex conflicts and edge conflicts are disallowed in *MAPFM*, and the *sum of individual costs* is optimised (as defined in [7]).

## II. Prior work

To solve *MAPF*, a number of algorithms have been proposed. Some of these algorithms are derived from A*. For example, ***A**** **with independence detection and operator decomposition** (*A*+ID+OD*) [6], *M** [5] and **enhanced partial expansion A*** (*EPEA**) [8] [9]. However, there are also algorithms which are not simply extending A* to avoid collisions. **Conflict based search** (*CBS*) [10] and **Increasing Cost Tree Search** (*ICTS*) are some of these.

This research is part of a set of parallel studies on how to extend all of the algorithms just discussed to give them the ability to do matching. Before this parallel research, a separate study has been performed [11] for a problem they call target-assignment and path-finding (*TAPF*). The difference between *TAPF* and *MAPFM* is that *TAPF* optimises the *makespan* instead of the *sum of individual costs*. To solve this [11] uses conflict based search, with a max-flow based algorithm to solve matchings within one team. It is yet unclear if it is possible to use such a max-flow based algorithm for *MAPFM*, however this is not a question that will be answered in this paper.

Solving *TAPF* and *MAPFM* with other algorithms has not yet been explored.

## III. A description of *M**

M* [5] is an algorithm to solve *MAPF*. To do so, it uses a form of independence detection. Each agent constantly follows an individually optimal path towards their own goal. While doing so, each agent keeps track of two things. The first is a collision set which stores all agents colliding with this agent at a particular time step. Each agent also has a backpropagation set, which is a set of neighbouring configurations used to get to this node.

*M** searches through states consisting of the position of each agent, together with their backprogpagation set and collision set. *A** is used to search through this search space. However, sometimes *A** will expand a state in which one or more collision occurs. This colliding state is then not added to the *A** search queue. Instead all states in the backpropagation set of this colliding state are re-added recursively to the *A** search queue to be re-expanded. This is done to find the shortest path for all agents around this collision.

[5] proves that *M** provides optimal solutions to *MAPF*.

## IV. *M** and matching

To add matching to *M**, this paper proposes two options which are proposed to be named "inmatching" and "prematching". In this section, both are explained and their advantages and disadvantages are discussed.

### A. Inmatching

*Inmatching* is the process of doing matching as a part of the pathfinding alorithm that is used. To understand it, it is useful to first look at inmatching in *A**. With *A**, the expansion of a state are all combinations of moves for all agents. *A** searches through the search space, until the goal state is removed from the frontier. Given an admissible heuristic *A** will guarantee that following the children of this first goal state gives a shortest path.

With *inmatching*, there is not one goal state. Instead any state in which all agents are on a goal of their color is considered a goal state. This means there are multiple goal states. To keep the heuristic admissible, the distance to the nearest goal state is used as a heuristic.

However, even though *inmatching* is quite a straight forward way to add matching to *A**, it doesn't work very well for *M**. The reason for this is that in *M** each agent tries to follow an individually optimal path. This means the next state is just the next position in the individually optimal path of each agent.

But, with *inmatching* there is not one individually optimal path. In stead, each agent needs to consider the shortest path to each of the possible goal positions. This means that the number of states that are expanded each timestep is bounded by the following formula:

$$\prod_{n=1}^{\#teams} \#goals_{team\ n}$$

On maps which contain few walls, the actual number of expanded states per timestep frequently comes close to this number leading to memory issues and decreased performance as can be seen in figure 1.

### B. Prematching

Alternatively, there is *prematching*. With *prematching* the *MAPFM* problem is transformed in a number of *MAPF* problems. Each possible matching is calculated in advance, and normal *M** as described by [5] is performed on each matching. In figure 1 it can be seen that in multiple scenarios *prematching* outperforms *inmatching*

Figure 1: Comparison between *prematch* and *inmatch* *M**

An algorithm to find these matchings

## V. Extensions to M*

### A. Recursive M*

### B. Distance matrices

### C. Operator decomposition

### D. Collision avoidance tables

### E. Matching Pruning

## VI. Other algorithms

## VII. Responsible research

## VIII. Conclusion

## References


[1] J. Mulderij, B. Huisman, D. Tönissen, K. van der Linden, and M. de Weerdt, “Train unit shunting and servicing: A real-life application of multi-agent path finding,” *arXiv preprint arXiv:2006.10422*, 2020.

[2] J. Li, A. Tinka, S. Kiesel, J. W. Durham, T. Kumar, and S. Koenig, “Lifelong multi-agent path finding in large-scale warehouses,” *arXiv preprint arXiv:2005.07371*, 2020.

[3] A. Mahdavi and M. Carvalho, “Distributed coordination of autonomous guided vehicles in multi-agent systems with shared resources,” in *2019 SoutheastCon*, IEEE, 2019, pp. 1–7.

[4] J. E. Hopcroft, J. T. Schwartz, and M. Sharir, “On the complexity of motion planning for multiple independent objects; pspace-hardness of the" warehouseman’s problem",” *The International Journal of Robotics Research*, vol. 3, no. 4, pp. 76–88, 1984.

[5] G. Wagner and H. Choset, “M*: A complete multirobot path planning algorithm with performance bounds,” in *2011 IEEE/RSJ international conference on intelligent robots and systems*, IEEE, 2011, pp. 3260–3267.

[6] T. Standley, “Finding optimal solutions to cooperative pathfinding problems,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 24, 2010.

[7] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. Kumar, *et al.*, “Multi-agent pathfinding: Definitions, variants, and benchmarks,” *arXiv preprint arXiv:1906.08291*, 2019.

[8] M. Goldenberg, A. Felner, R. Stern, G. Sharon, N. Sturtevant, R. C. Holte, and J. Schaeffer, “Enhanced partial expansion a*,” *Journal of Artificial Intelligence Research*, vol. 50, pp. 141–187, 2014, ISSN: 10769757. DOI: `10.1613/jair.4171`.

[9] A. Felner, M. Goldenberg, G. Sharon, R. Stern, T. Beja, N. Sturtevant, J. Schaeffer, and R. C. Holte, “Partial-expansion a* with selective node generation,” *Proceedings of the 5th Annual Symposium on Combinatorial Search, SoCS 2012*, pp. 180–181, 2012.

[10] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, “Conflict-based search for optimal multi-agent pathfinding,” *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[11] H. Ma and S. Koenig, “Optimal target assignment and path finding for teams of agents,” *arXiv preprint arXiv:1612.05693*, 2016.